BAB III

METODE PENELITIAN

3.1 STUDI LITELATUR

Pada tahap ini, penulis mencari bahan-bahan pustaka yang berhubungan dengan prototype, visual, desain, dll. Analisa masalah yang terdiri dari studi literature yang bertujuan untuk mempelajari kebutuhan dan minat dari masyarakat.

3.2 STUDI PUSTAKA

Pada metode ini penulis melakukan pencarian sumber referensi dari jurnal untuk nantinya akan dijadikan sebagai acuan dalam penyusunan laporan makalah dan pembangunan program, berikut review jurnal yang digunakan :

Judul	Pencarian Rute Terdekat Pada Labirin Menggunakan Metode A*
Jurnal	EECCIS
Volume & Halaman	Vol. 6, No. 2
Tahun	2012
Penulis	Rengga Dionata Putra, Ir. Muhammad Aswin, MT. Dan Waru Djuriatno, ST., MT.
Reviewer	Novita Sari Azizah
Tujuan penelitian	Membuat agen mampu mencari rute terpendek untuk mencapai finish.
Subjek penelitian	Agen sebagai NPC.
Metode penelitian	Algoritma A* adalah sebuah algoritma yang telah diperkaya dengan menerapkan suatu heuristik, algoritma ini membuang

	langkah-langkah yang tidak perlu dengan pertimbangan bahwa langkah-langkah yang dibuang sudah pasti merupakan langkah yang tidak akan mencapai solusi yang diinginkan
Faktor pengaruh	Faktor pengaruh dari rute terpendek adalah panjang dan banyaknya rute yang tersedia.
Alasan dilakukan penelitian ini	Proses dari manusia dalam menyelesaikan sebuah rute pencarian pada masalah pencarian ruang terdekat yang sederhana masih mampu diselesaikan dengan waktu yang cukup singkat, tetapi jika jumlah rute yang ada sudah sedemikian banyaknya, maka kita akan mengalami kesulitan dan akan memakan waktu yang lama untuk menyelesaikannya.
Langkah-langkah	A. Perancangan PerangkatLlunak Perangkat lunak yang akan dibuat menggunakan bahasa pemrogaman Visual Basic.NET 2010 dan sistem yang digunakan untuk membangun perangkat lunak ini dirancang dengan spesifikasi mampu melakukan hal-hal sebagai berikut: 1. merubah data masukkan yang berupa Larik menjadi bentuk labirin 2. menghitung dan mencari jalan tercepat pada labirin B. desain aplikasi Langkh pertama adalah menyiapkan larik sebagai inputan utama, setelah itu aplikasi akan memvisuaisasikan menjadi labirin untuk dihitung dan memili langkah hingga finis. C. Inisialisasi larik Pada tahap ini aplikasi akan menginisialisasi data masukan yang kita persiapkan sebelumnya, menjadi

	sebuah labirin yang utuh, proses yang teradi pada tahap ini adalah, membedakan kode tembok, kode jalan masuk, kode jalan keluar dan kode jalan. D. Pemilihan langkah Pada tahap pemilihan langkah ini, dilakukan setelah data Larik di inisialisasi dan menjadi labirin secara utuh. Proses ini dilakukan dengan cara membandingkan area sekitar agen ter lebih dahulu, yaitu : area atas, bawah, kanan dan kiri labirin. agen akan menginisialisasi area sekitar dan membandingkan antara tembok, jalan ataukah jalan buntu E. Perhitungn Langkah Perhitungan akan langkah dilakukan ketika agen berada pada posisi awal (jalan masuk) hal ini bertujuan agar agen sudah bisa memilih langkah pada awal posisi. Pada aplikasi pencari rute terdekat ini menggunakan pytagoras untuk menghitung jarak. Perhitungan jarak dilakukan dengan menghitung jarak atar koordinat saat ini (x,y) dan finish (x,y). Perhitungan terus dilakukan ketika agen bergerak hingga agen menemukan jalan keluar/finish Kemudin agen menghitung area sekitar meliputi atas, bawah, kanan, kiri (bila memungkinkan) dengangaris finish menggunakanmetde pytagoras, sehingga didapat hasil obt yang paling ringan. F. Back track Proses backtrack ini hanya digunakan untuk labirin yang mempunyai jalan buntu atau rumit sehingga agen bisa bergerak mundur dan memilih jalan yang memberikan pilihan untuk jalan. Kemudian agen beada pada jalan buntu da akan kembali ke persimpangan yang memberi pilihan untuk jalan. Tanda “X” berarti jalan suah dilalui dan tak boleh dilalui oleh

	agen lagi sehingga agen akan bergerak kebawah.
Hasil penelitian	A. Labirin tanpa cabang labirin yang hanya mempunyai satu jalan atau labirin yang tidak memberikan agen pilihan jalan untuk dipilih. Agen berhasil menemukan jalan keluar/finish dengan 29 langkah. B. Labirin Bercabang Labirin bercabang adalah labirin yang mempunyai jalan lebih dari satu, sehingga ada beberapa kemungkinan mencapai jalan keluar, agen pun diberikan pilihan untuk memilih jalan. Pada labirin bercabang ini agen memerlukan 56 langkah untuk mencapai jalan keluar C. Labirin Buntu Bersolusi Labirin buntu bersolusi adalah labirin yang mempunyai jalan buntu, bercabang tapi tetap mempunyai solusi, dalam artian agen masih bisa mencapai jalan keluar/finish Pada labirin buntu bersolusi ini agen memerlukan 36 langkah untuk menemukan jalan keluar, pada gambar labirin diatas agen melakukan proses backtrack, proses ini di tandai dengan jalan berwarna biru pada gambar D. Labiri Buntu Labirin buntu adalah abirin yang tak mempunyai jalan keluar. Pada labirin ini agen melakukan backtrack, dan memerlukan 12 langkah untuk kembali ke posisi awal, karena tak mendapatkan solusi. E. Labirin 20 x 20 Bercabang, Buntu Bersolusi Labirin 20 x 20 adalah labirin yang mempunyai panjang 20 kotak dan lebar 20 kotak. Labirin ini mempunyai percabangan dan jalan buntu sehingga lebih rumit dari labirin sebelumnya. Agen melakukan 62 langkah untuk menemukan jalan keluar

Kelebihan	Pada metode ini dari semua rute yang ada A* hanya menghitung area yang dilalui saja. Area yang tidak dilalui di abaikan.
Kekurangan	A* tidak menjamin selalu mendapatkan rute terbaik (bobot terkecil)

Judul	IMPLEMENTASI DAN ANALISIS ALGORITMA A*(STAR) UNTUK MENENTUKAN JALUR DENGAN MULTIPLE GOAL PADA PERGERAKAN NPC (NON-PLAYABLE CHARACTER)
Jurnal	e-Proceeding of Engineering
Volume & Halaman	Vol.2, No.3
Tahun	2015
Penulis	Pratama Juliantono Taufiq, Agung Toto Wibowo, ST., MT., Gia Septiana, SSi., MSc.
Reviewer	Novita Sari Azizah
Tujuan penelitian	Menerapkan algoritma A* pada NPC simulasi game agar NPC bisa menentukan jalur secara otomatis dengan lebih dari satu tujuan karena umumnya implementasi algoritma ini hanya untuk satu tujuan.
Subjek penelitian	NPC
Metode penelitian	A* adalah algoritma Best First Search yang merupakan perpaduan Uniform Cost Search yang memilih jarak paling kecil dari simpul awal ke simpul berikutnya dan Greedy-Best First Search yang menggunakan nilai heuristik atau nilai perkiraan untuk menentukan simpul berikutnya.
Faktor pengaruh	Memiliki lebih dari satu tempat yang akan di tuju
Alasan dilakukan	Karakter yang digerakkan secara otomatis oleh komputer.

penelitian ini	Kondisi tersebut memunculkan kebutuhan akan NPC yang pintar dan bisa bersaing layaknya manusia yang emainkannya karena turn-based strategy game merupakan permainan dimana karakter yang dimainkan bergerak mendekati lawan dengan berbagai strategi atau perhitungan untuk mendapatkan suatu keputusan. Semakin pintar NPC yang dilawan oleh pemain, maka pemain akan terus mendapatkan tantangan dalam bermain game tersebut.
Langkah-langkah	Karena pergerakan karakter pemain tergantung pada pilihan dari pemain itu sendiri, maka digunakan algoritma A* biasa yang hanya memiliki satu tujuan, yaitu titik asal adalah tempat karakter berada dan titik akhir adalah lokasi pada map yang dipilih oleh pemain. 3.1.1 Inisialisasi Awal Pada inisialisasi awal ini dilakukan pemberian nilai awal pada setiap variable yang digunakan pada algoritma tersebut, diantaranya adalah - Array Open yang diisi dengan start node - Array Closed berupa array kosong 3.1.2 Isi Properti Dari Node Suksesor Setiap node memiliki properti sebagai indentitas node tersebut. Adapun properti yang dimiliki oleh tiap node adalah sebagai berikut: - Parent yang merupakan bestNode sebelumnya - Value yang merupakan nilai dari node tersebut yang nilainya diambil dari penghitungan alamat koordinat node tersebut - X, alamat node pada sumbu x - Y, alamat node pada sumbu y - f, nilai f-cost yang dihitung dari node tersebut ke node tujuan dengan menggabungkan nilai heuristik dan nilai g

	- g, nilai yang di dapat dari jarak node tersebut dengan node asal melalui parent sebelumnya 3.2 Perhitungan Fungsi Heuristik Penghitungan nilai heuristik untuk kasus multiple destination sedikit berbeda dengan penghitungan pada algoritma A* biasanya yang memakai Manhattan Distance, Diagonal Distance, ataupun Euclidean Distance. Dimulai dengan menghitung jarak antar goal dengan metode Euclidean Distance, kemudian masing-masing jarak tersebut ditambahkan dengan metode permutasi dari goal-goal yang ada sehingga menghasilkan susunan arah yang memiliki total jarak minimum. Setelah mendapatkan masing-masing susunannya sesuai dengna masing-masing goal, ditambahkan dengan jarak garis lurus dari node suksesor ke masing-masing goal. Setelah itu dipilih jarak yang paling kecil untuk di jadikan nilai heuristik. Untuk menentukan mana jalur yang selanjutkan akan dipilih adalah jalur yang berada di array Open dan memiliki f-cost minimum. Penghitungan f-cost dilakukan dengan mengkombinasikan jarak garis lurus. Kombinasi minimumlah yang menjadi f-cost dari node tersebut, seperti contoh pada gambar, garis A yang memiliki panjang 3,606 di tambahkan dengan kombinasi minimum dari jarak masing-masing node tujuan yang dimulai dari tujuan yang dituju oleh garis A yaitu garis b ditambah c dengan total 7,212. Sehingga f-cost untuk node pada koordinat (5,2) adalah garis A + garis b + garis c dengan total 10,818.
Hasil penelitian	Dengan menggunakan kombinasi jarak antar goal dari masing-masing tujuan, maka algoritma A*(Star) dapat menghasilkan jalur yang complete dan optimal pada kasus Multiple Destination. Sehingga NPC dapat menentukan jalur

	mana yang akan di lalui untuk mencapai seluruh tujuannya.
Kelebihan	Mampu mencari jalur terpendek dengan beberapa tujuan, sehingga tidak hanya ada satu tujuan yang dituju.
Kekurangan	Penunjang dari hasil tersebut adalah nilai heuristik yang didapat akibat pengaruh dari tujuan lain, bukan seperti Euclidean atau Manhattan Distance yang hanya mengandalkan acuan satu tujuan.

Hapsari Tilawah dengan judul "Penerapan Algoritma A-star (A*) Untuk Menyelesaikan Masalah Maze", jurnal oleh Riwinoto dan Alfian dengan judul "Implementasi *Pathfinding* dengan Algoritma A* pada *Game* Funny English Menggunakan Unity 3D Berbasis Graf Navmesh", jurnal oleh Andy Pramono, S.Kom, MT. dengan judul "Algoritma *Pathfinding* A* pada *Game* RPG Tanaman Higienis", jurnal oleh Ibnu Zakifardan dengan judul "Implementasi Algoritma Dynamic Weighthing A* Untuk Pencarian Rute Terpendek Pada NPC Dan Fisher-Yates Shuffle Untuk Pengaturan Konten Pada *Game* 3D Finding Diamond", jurnal oleh Sukiman, Jeffrey, dengan judul "Simulasi Pencarian Shortest Distance Path". Pembelajaran dari berbagai macam literatur dan dokumen sangat menunjang pengerjaan dan membantu guna kelancaran Tugas Akhir Skripsi ini.

3.3 PENGEMBANGAN *GAME*

Pengembangan *game* ini menggunakan prosedur pengembangan program multimedia, berikut tahapan-tahapan dari prosedur pengembangannya :

1. Perancangan Konsep

   Langkah awal membangun aplikasi ini adalah merancang konsep dari *game*. Pada langkah ini ditentukan *genre game* dengan mengimplementasikan *pathfinding* serta melakukan *collecting material* merupakan tahapan yang digunakan untuk mengumpulkan bahan-bahan yang dibutuhkan dalam pembuatan *game*.

2. Perancangan Desain

   Setelah tema dan konsep ditemukan, kemudian membuat skenario atau membuat alur cerita. Pada percancangan desain dilakukan dengan pembuatan *storyboard*.

3. Pembuatan *Game* (*Development*)

   Pada tahap ini *game* dibangun dengan mengaplikasikan konsep perancangan konsep dan desain ke dalam aplikasi atau root yang akan digunakan dalam proses pembuatan *game.* Berikut beberapa proses dalam *development game* :

   a) Modeling

   Pada tahap ini dilakukan pembuatan objek 2D sesuai sketsa / rancangan sebelumnya. Pembuatan objek 2D dirancang dengan menggunakan software adobe photoshop dan unity 3D untuk desain *game*.

   b) Animation

   Animation merupakan proses pembuatan animasi untuk model. Arah dimulainya suatu gerakan animasi dapat disesuaikan dengan *storyboard* yang telah dibuat pada tahap perancangan desain.

c) *Pathfinding*

Pada tahap ini dibuat lintasan atau *pathfinding* yang akan digunakan untuk arena permainan.

d) *Game*

Ditahap ini akan dilakukan *coding source code* untuk physicnya serta *engine* secara keseluruhan *engine* untuk fisika (*physic*), *engine* untuk suara 3D, engine untuk *AI*, dsb.

4. Pengujian *Black Box*

Pengujian *game* dilakukan dengan metode *black box.* Pengujian *black box* adalah pengujian aspek fundamental sistem tanpa memperhatikan struktur logika internal perangkat lunak. Dengan metode ini dapat diketahui apakah perangkat lunak berfungsi dengan benar seperti : unit, integrasi, fungsional, sistem dan penerimaan. Pengujian *black box* merupakan metode perancangan data uji yang didasarkan pada spesifikasi perangkat lunak. Data uji dieksekusi pada perangkat lunak di cek apakah sudah sesuai dengan yang di harapkan.